NONLINEAR OPTIMIZATION

Qingsha Cheng 程庆沙

Surrogate-Based Modeling and Optimization II: Functional Surrogate Models

Classification and examples of surrogate models

Functional versus physical surrogate models

Regression models

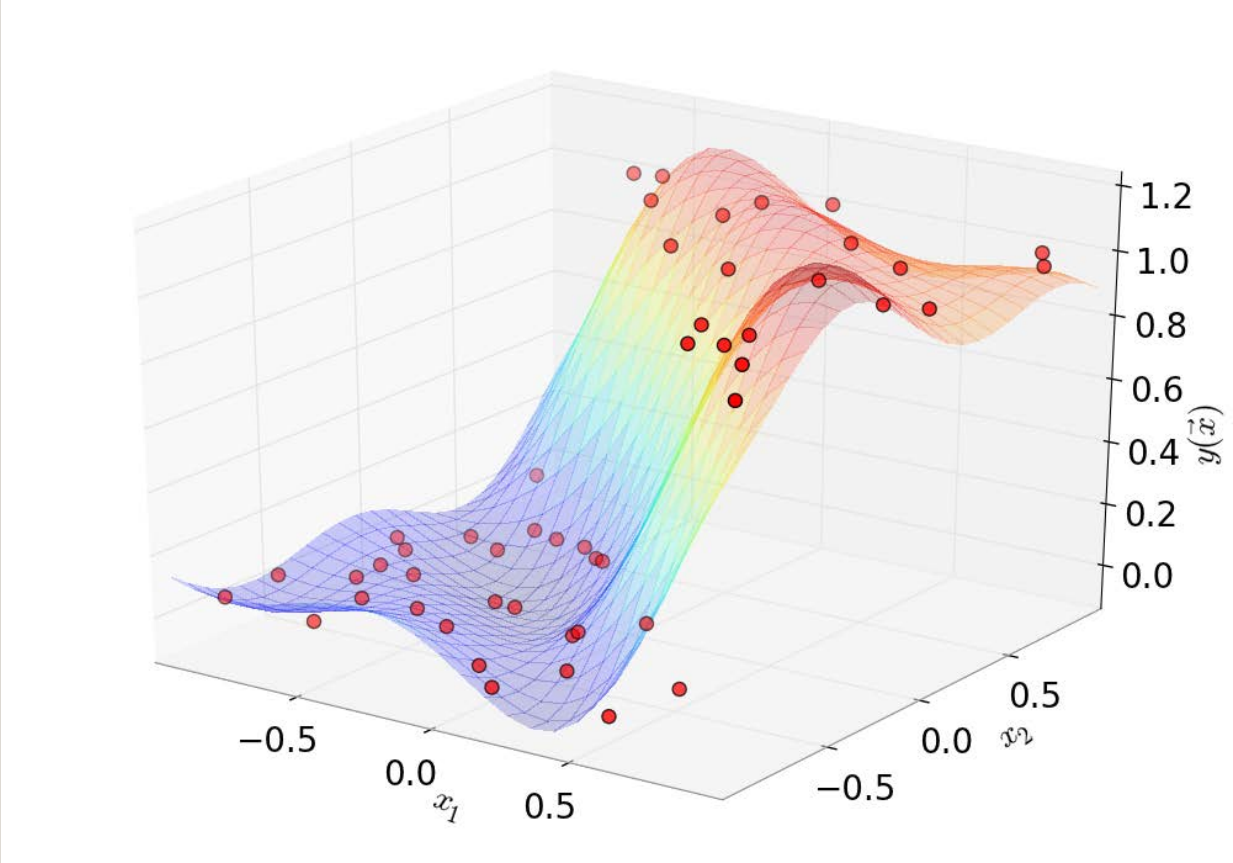
Radial basis functions

Kriging

Neural networks

Surrogate Modeling

Overall goal: given a function $f : X \to R$, $X \subseteq R^n$, create a surrogate model $s : X \to R$ so that s is representation of f over X as good as possible in a given sense.



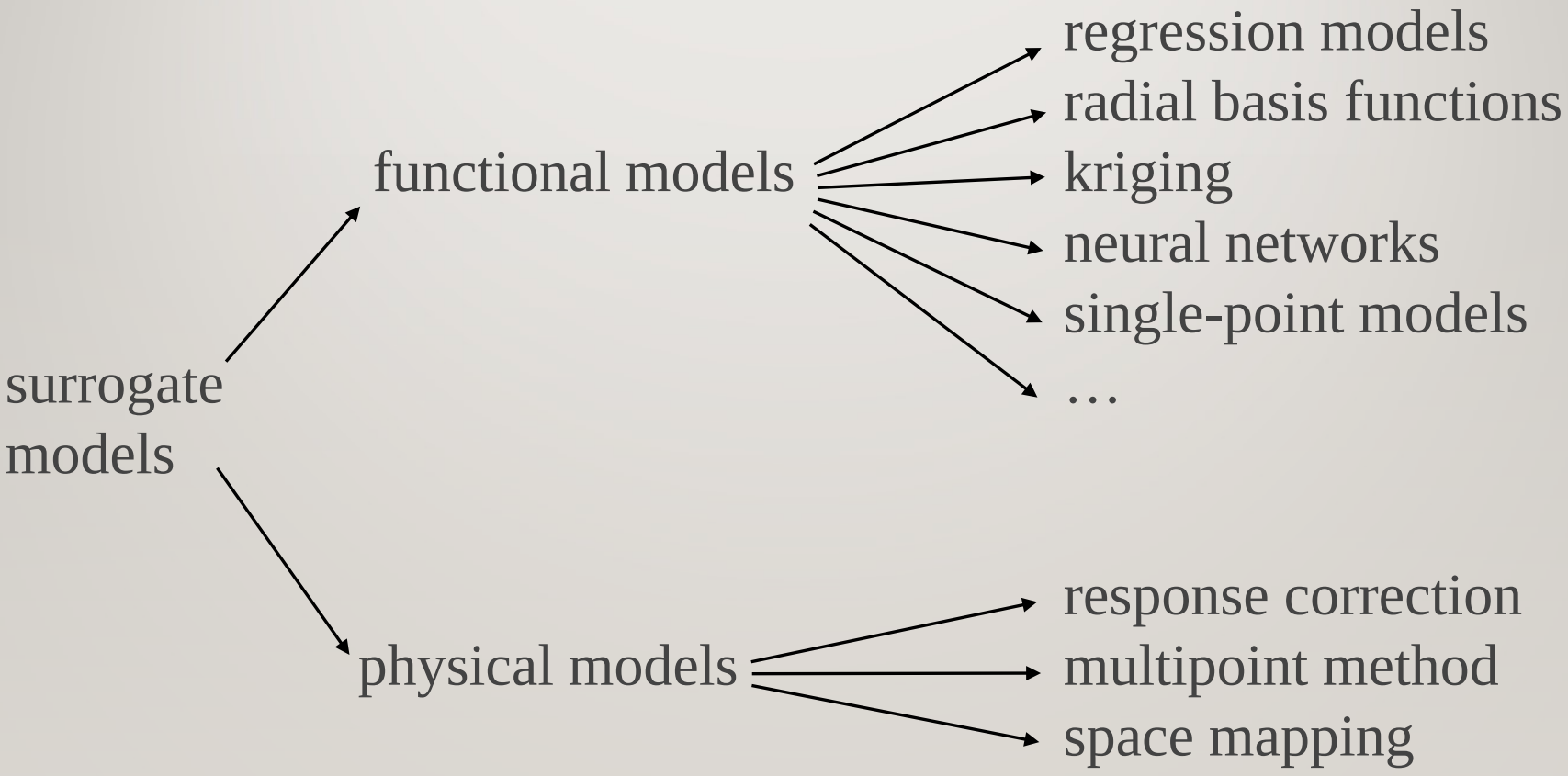
Surrogate Modeling Properties

Depending on application, we expect the surrogate model to have certain properties, such as:

1. Surrogate model s is computationally cheaper than f.
2. Surrogate model s has better analytical properties than f (continuity, differentiability)

Remark: generalization of the concepts described in this lecture to functions $f : X \rightarrow R^m$ is straightforward.

Classification of Surrogate Models



Functional Surrogate Models

Constructed **without** any particular knowledge of **the physical system**

Based on **algebraic expressions** and **empirical data**

Exist based on the sampled data obtained from a physical system

Generic => applicable to a wide class of problems

Typically, very cheap to evaluate

Typically, require considerable amount of data from a physical system

Physical Surrogate Models

Based on **knowledge** about the particular **physical system** in question

Evaluation of a physical model may involve, e.g., numerical solution of differential equations or even actual measurements of a physical system

Exist independently of the physical system

Dedicated => applicable to a given problem (reuse across different problems is rare)

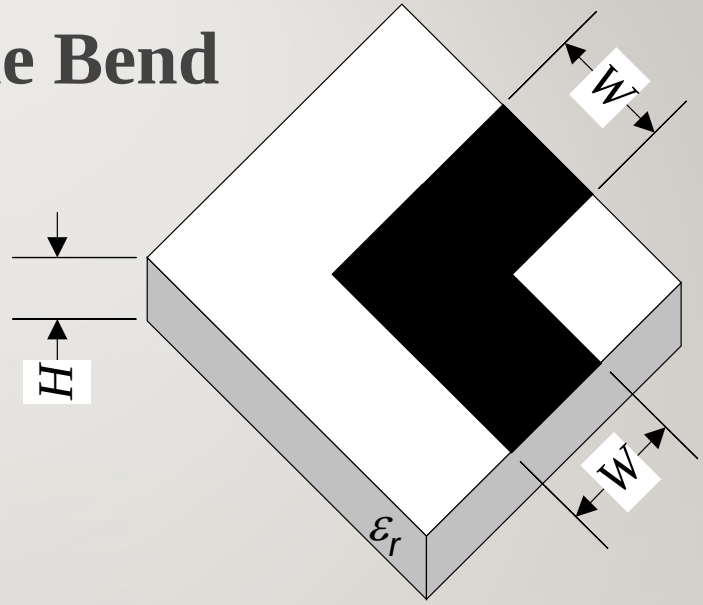
Typically, more expensive to evaluate than functional models

Typically, require no data from a physical system

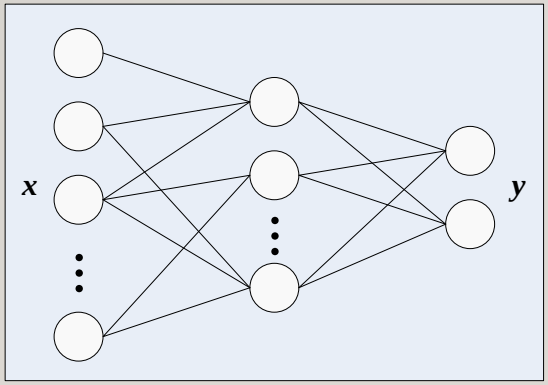
Example: Microstrip Right-Angle Bend

(*Bandler et al., 2001*)

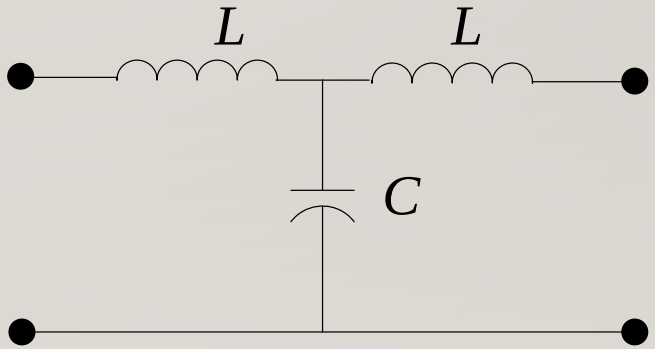
Physical system: microstrip structure simulated using *Sonnet* ***em***



Functional model:
neural network

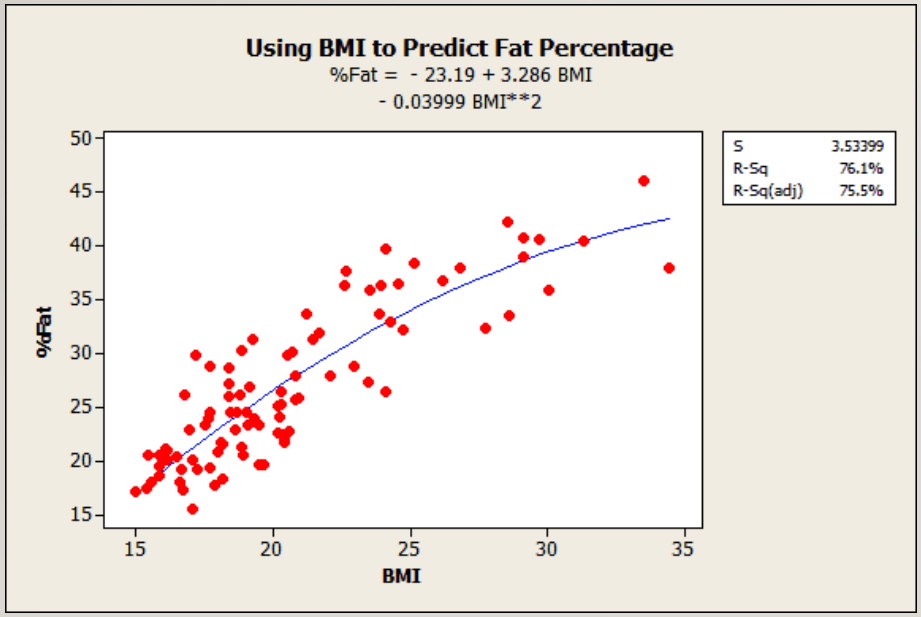


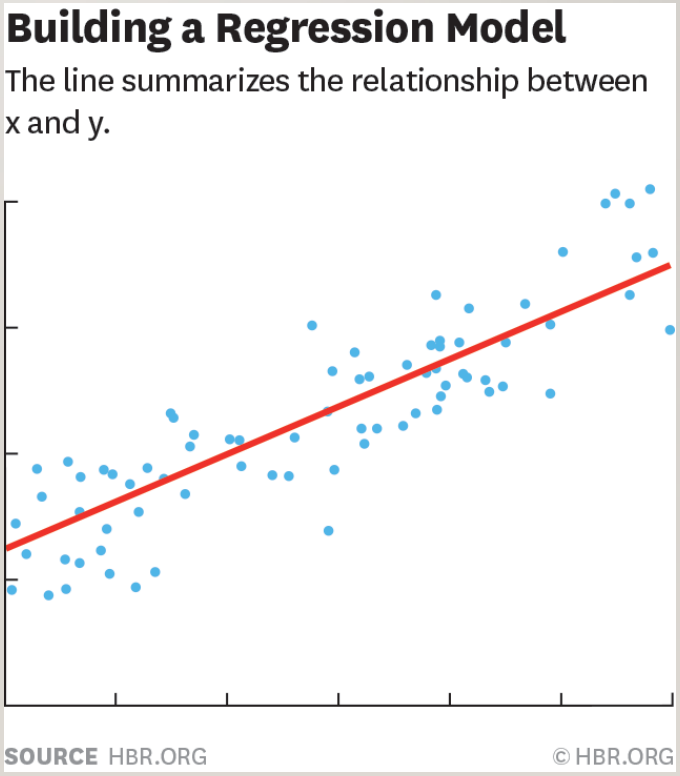
Physical model:
equivalent circuit



Regression Models

Regression is a process of fitting a model represented by the function $s(x,\lambda)$, to the pre-sampled data $(t^{(i)},y_i)$, $i = 1, \ldots, N$, of the response function from the original model f (i.e., $y_i = f(t^{(i)})$)





Regression Models

In particular, we want to find the values of parameters λ that solve the following regression (data fitting) problem

$$\min_{\lambda} \sum_{i=1}^{N} \left(y_i - s(t^{(i)}, \lambda) \right)^2$$

Function $s(x,\lambda)$ is called a regression model; s may be a nonlinear function of both x and λ. Therefore, model parameters λ are obtained using nonlinear optimization procedures in general.

Regression Models: Linear Regression

Linear regression models:

$$s(x,\lambda) = \lambda^T v(x)$$

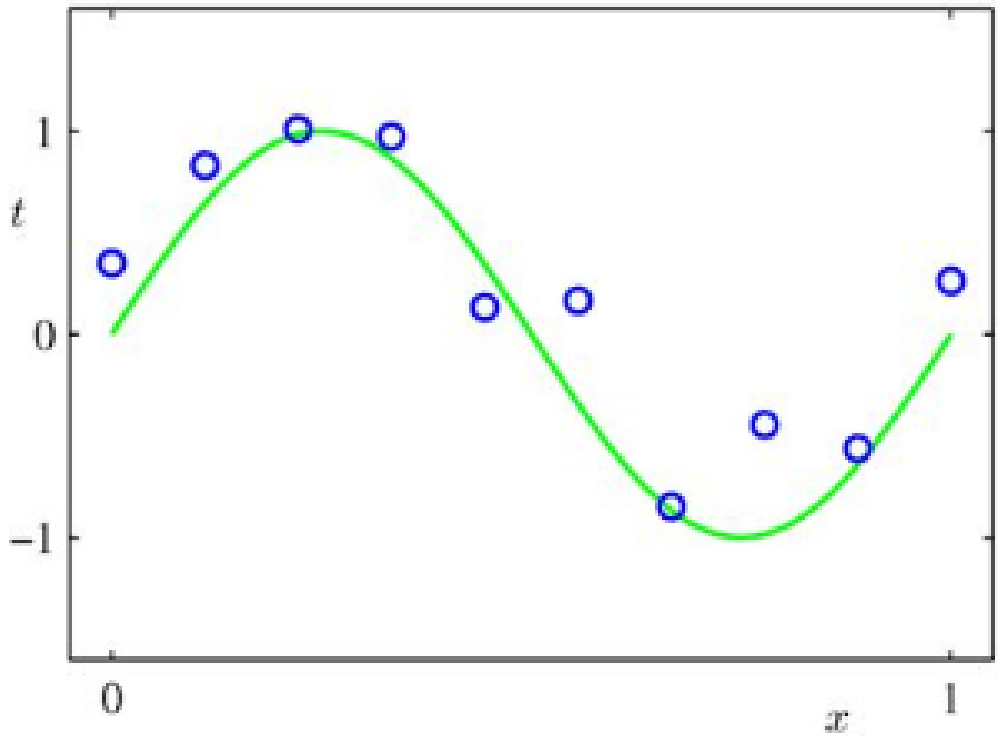
where $\lambda = [\lambda_1 \ldots \lambda_u]^T$, and $v : R^n \rightarrow R^u$ is a vector with u basis functions $v_j(x)$, $j = 1, \ldots, u$

Linear regression is an important case because parameters λ can be found analytically

Remark: note that the word “linear” refers to model parameters λ only: basis functions v may be nonlinear w.r.t. x.

Regression Models: Linear Regression

Example: Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Regression Models: Linear Regression

Coefficients λ can be found by solving a linear system

$$X\lambda = y$$

where

$$X = \begin{bmatrix} v_1(t^{(1)}) & v_2(t^{(1)}) & \cdots & v_u(t^{(1)}) \\ v_1(t^{(2)}) & v_2(t^{(2)}) & \cdots & v_u(t^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ v_1(t^{(N)}) & v_2(t^{(N)}) & \cdots & v_u(t^{(N)}) \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} f(t^{(1)}) \\ f(t^{(2)}) \\ \vdots \\ f(t^{(N)}) \end{bmatrix}$$

Remark: rank of matrix X must be equal u in order to ensure existence of unique λ. In particular, we have to have $N \geq u$

Regression Models: Linear Regression

the least-square solution of the previous problem is found by

$$\lambda = (X^T X)^{-1} X^T y$$

where

$$X = \begin{bmatrix} v_1(t^{(1)}) & v_2(t^{(1)}) & \cdots & v_u(t^{(1)}) \\ v_1(t^{(2)}) & v_2(t^{(2)}) & \cdots & v_u(t^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ v_1(t^{(N)}) & v_2(t^{(N)}) & \cdots & v_u(t^{(N)}) \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} f(t^{(1)}) \\ f(t^{(2)}) \\ \vdots \\ f(t^{(N)}) \end{bmatrix}$$

Regression Models: Linear Regression with Linear Basis Functions

Basis functions $v_j : R^n \rightarrow R,\ j = 1, \ldots, n + 1$:

$$v_1(x) = 1$$
$$v_2(x) = x_1$$
$$\ldots$$
$$v_n(x) = x_{n\text{-}1}$$
$$v_{n+1}(x) = x_n$$

Surrogate model:

$$s(x) = \lambda_0 + \sum_{i=1}^{n} \lambda_i x_i$$

Coefficients λ can be found by solving a linear system as explained before

Regression Models: Linear Regression with Quadratic Basis Functions

Basis functions $v_j : R^n \rightarrow R$, $j = 1, \ldots, (n+2)(n+1)/2$:

$$v_1(x) = 1$$
$$v_2(x) = x_1$$
$$\ldots$$
$$v_{n+1}(x) = x_n$$
$$v_{n+2}(x) = x_1 x_1$$
$$v_{n+3}(x) = x_2 x_1$$
$$v_{n+4}(x) = x_2 x_2$$
$$v_{n+5}(x) = x_3 x_1$$
$$\ldots$$
$$v_{(n+2)(n+1)/2}(x) = x_n x_n$$

Surrogate model:

$$s(x) = \lambda_0 + \sum_{i=1}^{n} \lambda_i x_i + \sum_{i=1}^{n} \sum_{j \le i} \lambda_{ij} x_i x_j$$

Example: Linear Regression with Linear Basis Functions in One Dimension

We have data pairs $(t^{(i)},y_i)$, basis functions $\{1,x\}$, and linear surrogate model $s(x) = \lambda_0 + \lambda_1 x$

Main equation

$$\begin{bmatrix} 1 & t^{(1)} \\ 1 & t^{(2)} \\ \vdots & \vdots \\ 1 & t^{(N)} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Least-square solution

$$\lambda_0 = \frac{\sum_i (t^{(i)})^2 \sum_i y_i - \sum_i t^{(i)} \sum_i t^{(i)} y_i}{N \sum_i (t^{(i)})^2 - \left(\sum_i t^{(i)} \right)^2} \qquad \lambda_1 = \frac{N \sum_i t^{(i)} y_i - \sum_i t^{(i)} \sum_i y_i}{N \sum_i (t^{(i)})^2 - \left(\sum_i t^{(i)} \right)^2}$$

Example: Linear Regression with Quadratic Basis Functions in One Dimension

We have data pairs $(t^{(i)}, y_i)$ and basis functions $\{1, x, x^2\}$

Quadratic surrogate model

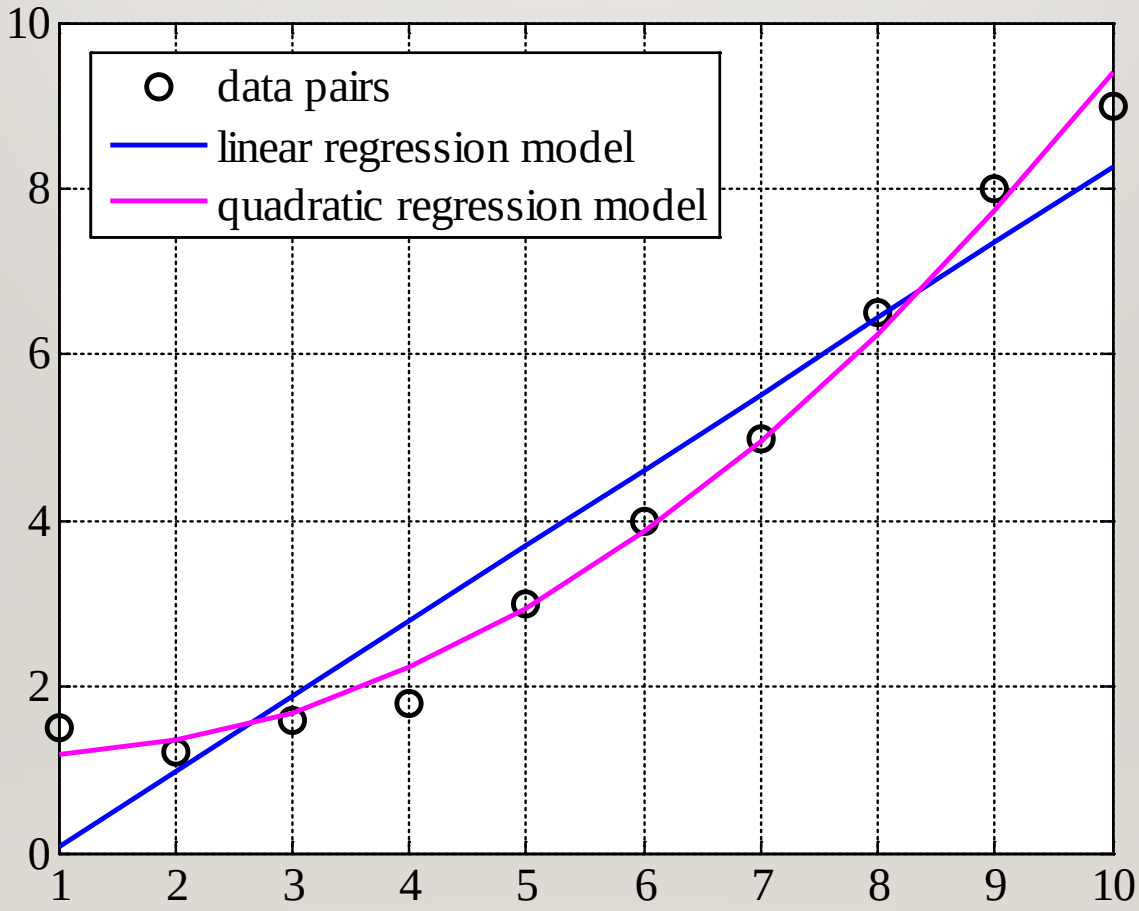
$$s(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2$$

Main equation

$$\begin{bmatrix} 1 & t^{(1)} & (t^{(1)})^2 \\ 1 & t^{(2)} & (t^{(2)})^2 \\ \vdots & \vdots & \vdots \\ 1 & t^{(N)} & (t^{(N)})^2 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Regression in One Dimension: Numeric Example

Data pairs:
(1,1.5), (2,1.2), (3,1.6), (4,1.8), (5,3), (6,4), (7,5), (8,6.5), (9,8), (10,9)



Maximally Flat Quadratic Approximation

Quadratic regression model has $(n+2)(n+1)/2$ coefficients => large number of data required to create the model for large n

Maximally flat quadratic approximation finds coefficients of a quadratic regression model so that its second order terms are as small as possible

Maximally Flat Quadratic Approximation

Suppose that $n + 1 < N < (n+1)(n+2)/2$ evaluations of the expensive model are performed: pre-sampled data $(t^{(i)}, y_i)$, $i = 1, \ldots, N$

coefficients for zero and first order term

$$\Lambda_1 = \left[\lambda_0 \ \lambda_1 \ \lambda_2 \ \ldots \lambda_n \right]^T$$

coefficients for second order term

$$\Lambda_2 = \left[\lambda_{11} \ \lambda_{22} \ \ldots \lambda_{nn} \ \ \lambda_{12} \ \ \lambda_{13} \ \ \ldots \ \ \lambda_{n-1,n} \right]^T$$

we need to solve

$$\min \left\| \Lambda_2 \right\|^2$$

$$s.t. \ Q\Lambda = \boldsymbol{Y}$$

Maximally Flat Quadratic Approximation

Set up a system of linear equations

$$\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \boldsymbol{Y}$$

where

$$\Lambda_1 = \left[\lambda_0 \ \lambda_1 \ \lambda_2 \ \ldots \lambda_n \right]^T \qquad \Lambda_2 = \left[\lambda_{11} \ \lambda_{22} \ \ldots \lambda_{nn} \ \ \lambda_{12} \ \ \lambda_{13} \ \ \ldots \ \ \lambda_{n-1,n} \right]^T$$

$$Y_1 = \left[y_1 \ \ y_2 \ \ \ldots \ \ y_{n+1} \right]^T \qquad Y_1 = \left[y_{n+2} \ \ y_{n+3} \ \ \ldots \ \ y_p \right]^T$$

while matrices Q_{11}, Q_{12}, Q_{21} and Q_{22} are determined by the coordinates of the base points

Maximally Flat Quadratic Approximation

Reduced system for vector Λ_2

$$C\Lambda_2 = \Gamma$$

where

$$C = Q_{22} - Q_{21}Q_{11}^{-1}Q_{12}$$

$$\Gamma = Y_2 - Q_{21}Q_{11}^{-1}Y_1$$

Vectors Λ_1 and Λ_2 are found as follows

$$\Lambda_2 = C^T \left(CC^T \right)^{-1} \Gamma$$

$$\Lambda_1 = Q_{11}^{-1}Y_1 - Q_{11}^{-1}Q_{12}\Lambda_2$$

Approximation/Interpolation with Higher-Order Polynomials

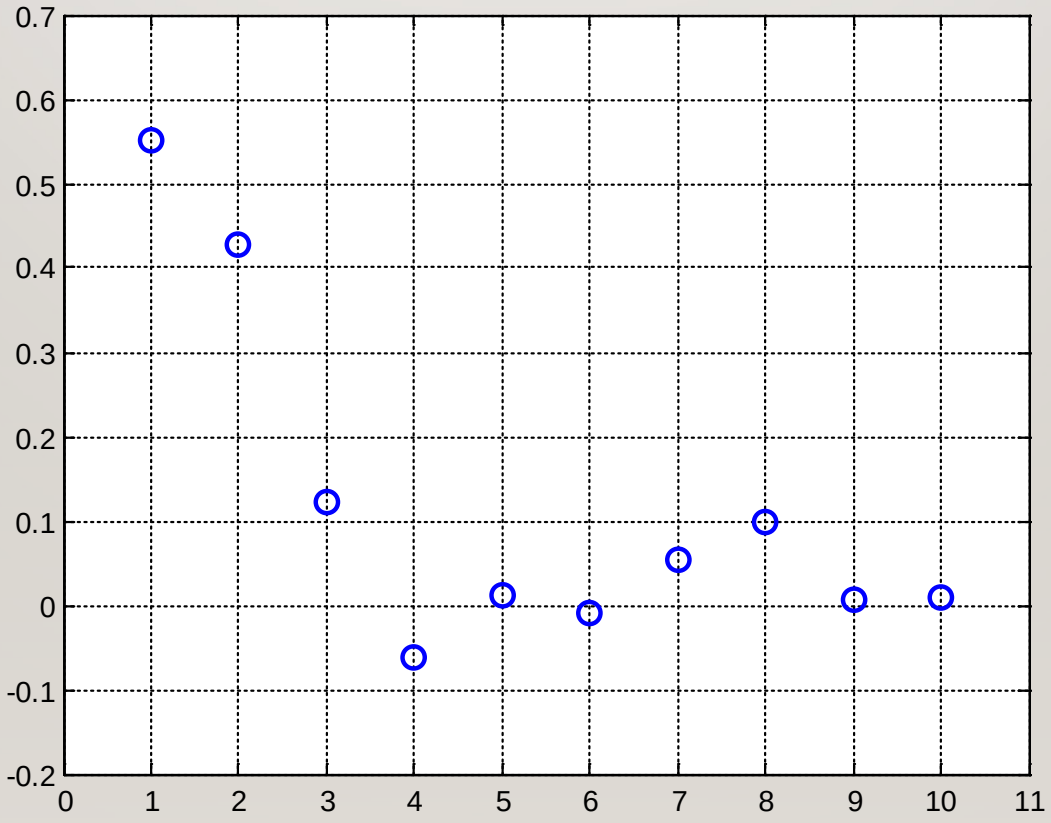
In practice, polynomial-based regression is performed using polynomials up to order three or four

Higher-order polynomials should not be used because:

1. Have poor generalization capability (i.e., prediction ability outside the region containing data pairs used to set up the model)
2. May exhibit undesired behavior such as under- and over-shooting
3. Require too many coefficients

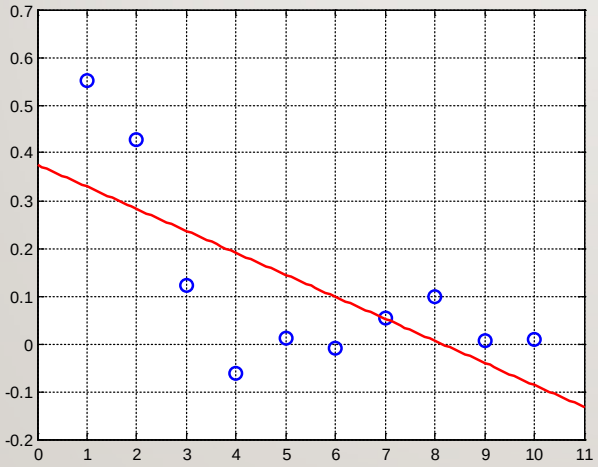
Approximation/Interpolation with Higher-Order Polynomials

Example: Approximate the following data pairs using polynomials

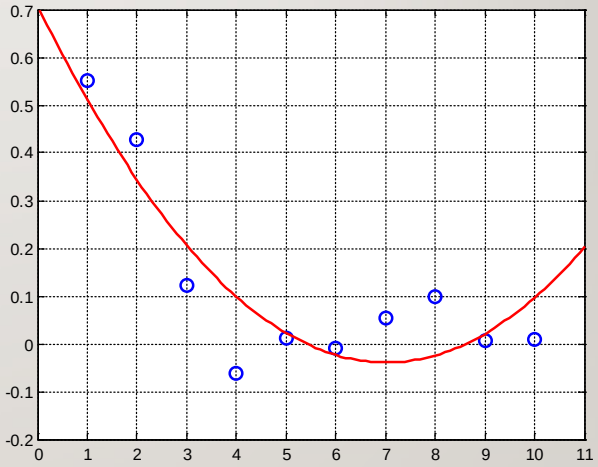


Approximation/Interpolation with Higher-Order Polynomials

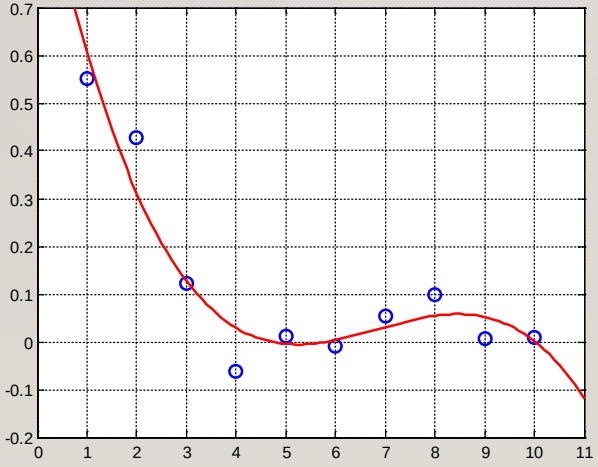
$n = 1$:



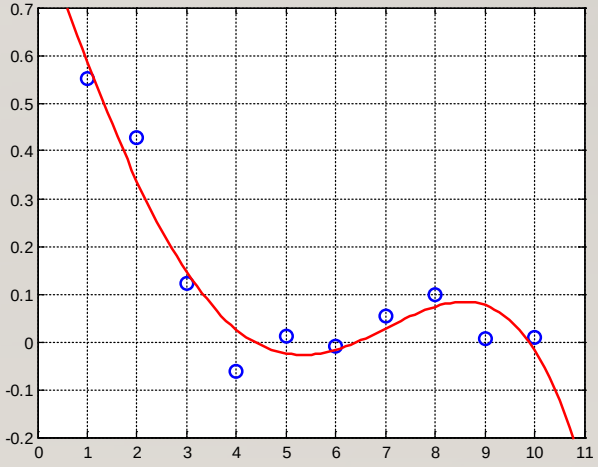
$n = 2$:



$n = 3$:

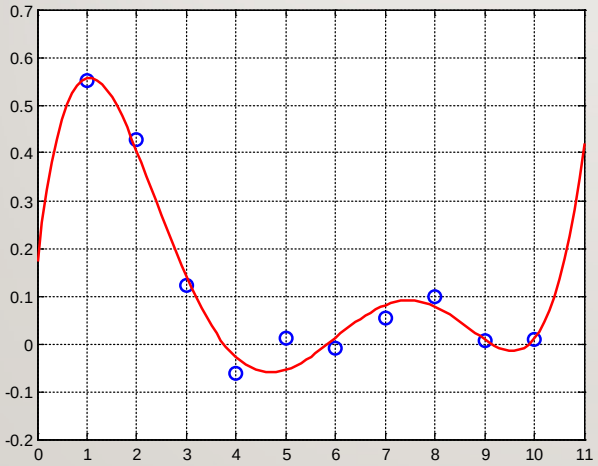


$n = 4$:

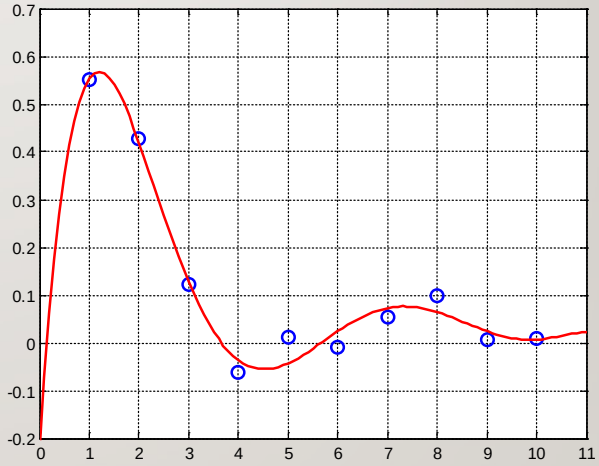


Approximation/Interpolation with Higher-Order Polynomials

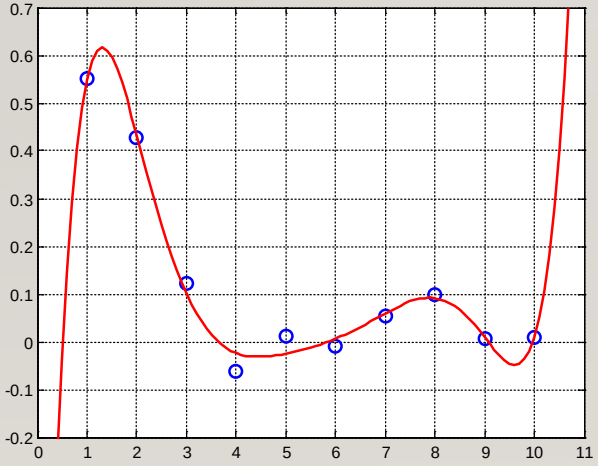
$n = 5$:



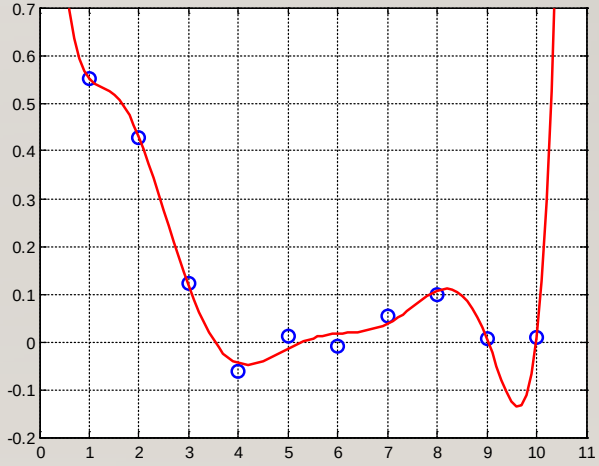
$n = 6$:



$n = 7$:

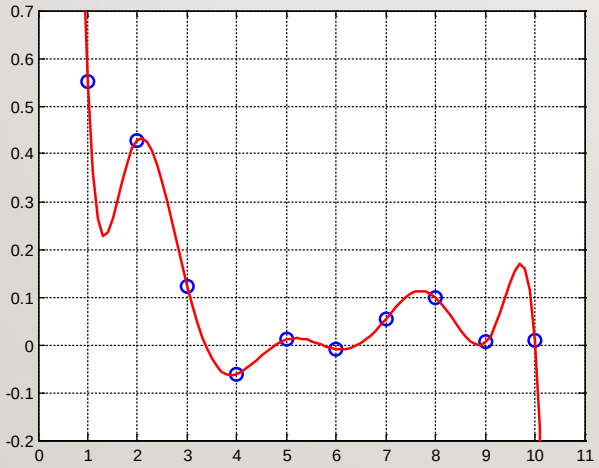


$n = 8$:



Approximation/Interpolation with Higher-Order Polynomials

$n = 9$:



Note that $n = 9$ is the highest-order polynomial that can be uniquely defined for the given data (10 samples) and it actually interpolates the data points.

Radial Basis Functions

Radial basis function interpolation uses linear combination of radially symmetric functions $\phi : R_+ \cup \{0\} \to R$, based on Euclidean distance, to approximate function f using the following surrogate model:

$$s(x) = \sum_{i=1}^{N} \lambda_i \phi(\| x - t^{(i)} \|)$$

where $(t^{(i)}, y_i)$, $i = 1, \ldots, N$, is pre-sampled data from the function f

Radial Basis Functions

We want the interpolation condition to be satisfied, i.e.,

$$s(t^{(j)}) = y_j$$

for all $j = 1, \ldots, N$.

This leads to the following condition on λ: $\Phi\lambda = y$

where

$$\Phi = \begin{bmatrix} \phi\left(\| t^{(1)} - t^{(1)} \|\right) & \phi\left(\| t^{(1)} - t^{(2)} \|\right) & \cdots & \phi\left(\| t^{(1)} - t^{(N)} \|\right) \\ \phi\left(\| t^{(2)} - t^{(1)} \|\right) & \phi\left(\| t^{(2)} - t^{(2)} \|\right) & \cdots & \phi\left(\| t^{(2)} - t^{(N)} \|\right) \\ \vdots & \vdots & \ddots & \vdots \\ \phi\left(\| t^{(N)} - t^{(1)} \|\right) & \phi\left(\| t^{(N)} - t^{(2)} \|\right) & \cdots & \phi\left(\| t^{(N)} - t^{(N)} \|\right) \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Radial Basis Functions: Examples

Gaussian $\phi(r) = e^{-cr^2}, \ r \geq 0, \ c > 0$

Inverse multiquadric $\phi(r) = (r^2 + c^2)^{-1/2}, \ r \geq 0, \ c > 0$

Linear $\phi(r) = r, \ r \geq 0$

Multiquadric $\phi(r) = (r^2 + c^2)^{1/2}, \ r \geq 0, \ c > 0$

Thin plate spline $\phi(r) = r^2 \log r, \ r \geq 0$

Cubic $\phi(r) = r^3, \ r \geq 0$

Radial Basis Functions: Example

Interpolate function $f(x) = \exp(-x/2)\cdot\sin(2x)$ in the interval [1,5] using gaussian RBF with $c = 1$

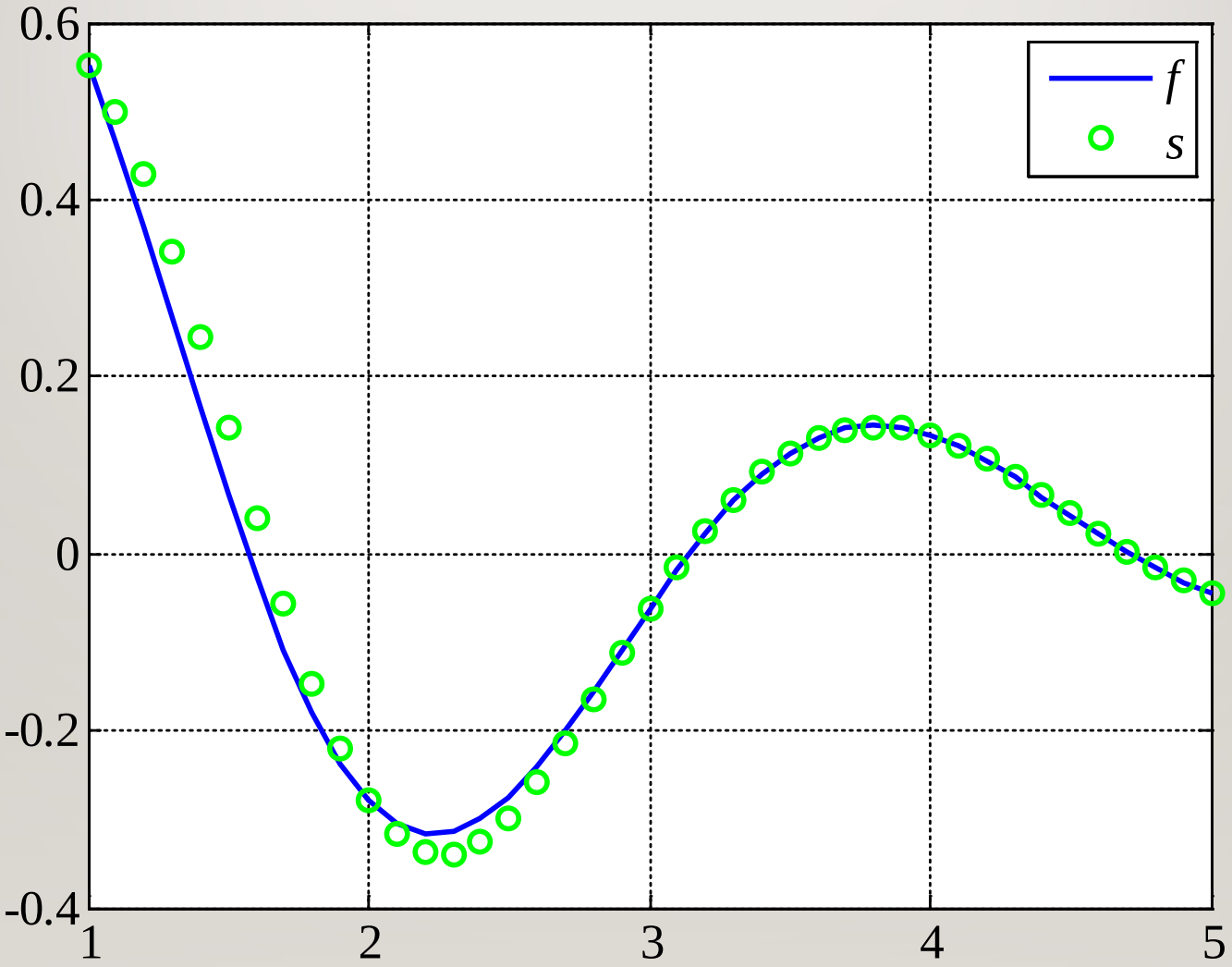
Data pairs: (1,0.5515), (2,-0.2784), (3,-0.0623), (4,0.1339), (5,-0.0447)

Main equation:

$$\begin{bmatrix} 1.00000 & 0.36788 & 0.01832 & 0.00012 & 0.00000 \\ 0.36788 & 1.00000 & 0.36788 & 0.01832 & 0.00012 \\ 0.01832 & 0.36788 & 1.00000 & 0.36788 & 0.01832 \\ 0.00012 & 0.01832 & 0.36788 & 1.00000 & 0.36788 \\ 0.00000 & 0.00012 & 0.01832 & 0.36788 & 1.00000 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{bmatrix} = \begin{bmatrix} +0.5515 \\ -0.2784 \\ -0.0623 \\ +0.1339 \\ -0.0447 \end{bmatrix}$$

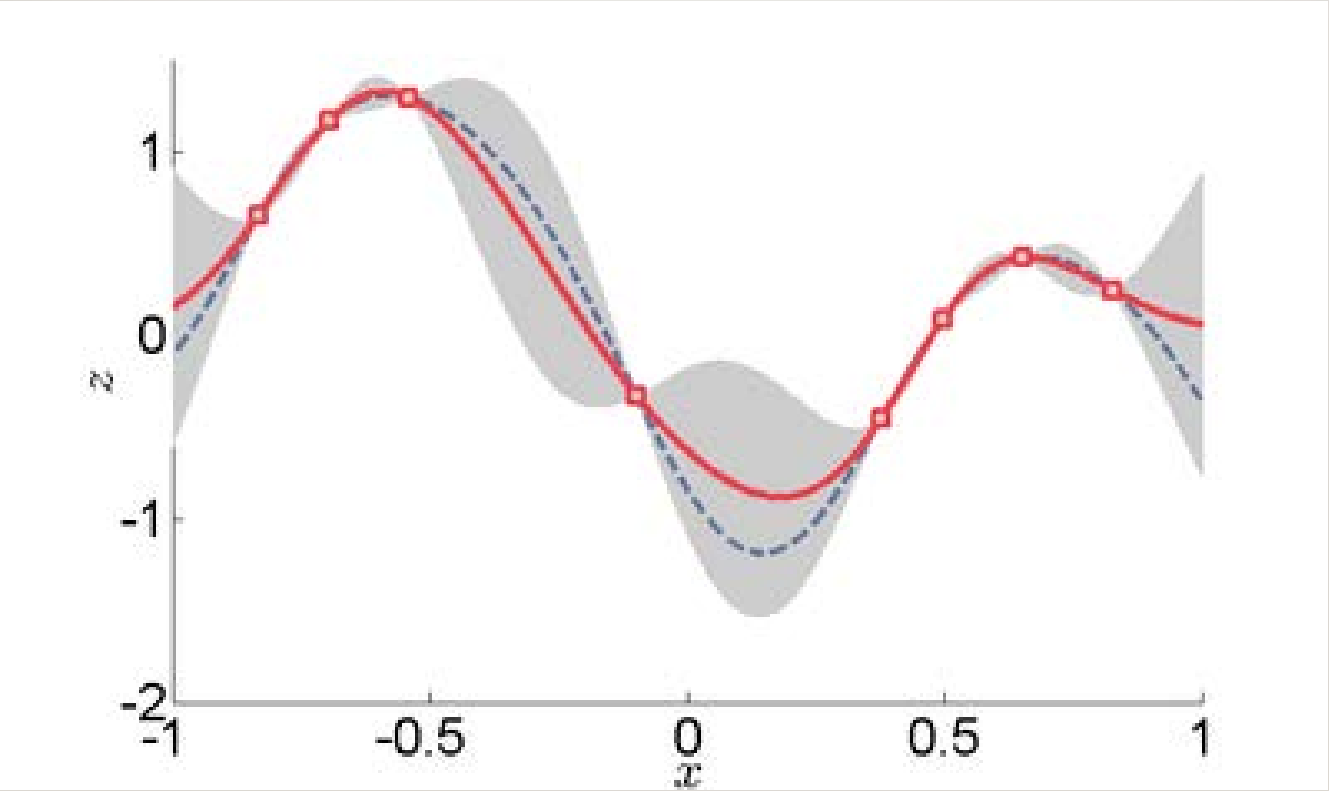
Coefficients: $\lambda = [0.7697 \;\; -0.5976 \;\; 0.0906 \;\; 0.1485 \;\; -0.1009]^T$

Radial Basis Functions: Example



Kriging

Kriging model is defined as the combination of a global trend function and a function that models departures from the trend



Example of one-dimensional data interpolation by kriging, with confidence intervals. Squares indicate the location of the data. The kriging interpolation, shown in red, runs along the means of the normally distributed confidence intervals shown in gray. The dashed curve shows a spline that is smooth, but departs significantly from the expected intermediate values given by those means.

Kriging

The trend function, $P(x)$, is usually a polynomial (e.g., constant, linear, or quadratic)

The local departures, $Z(x)$, typically use RBFs

The combined model interpolates N data points used to construct it.

The kriging surrogate model is given as

$$s(x) = P(x) + Z(x)$$

Kriging

The polynomial regression term is defined as $P(x) = \overline{\beta}^T q(x)$

where $\overline{\beta}$ are regression parameters $\overline{\beta} = \left[\overline{\beta}_1 \;\; \overline{\beta}_2 \;\; \ldots \;\; \overline{\beta}_r \right]^T$

and $q(x)$ is a vector of r functions from a polynomial basis

$$q(x) = \left[q_1(x) \;\; q_2(x) \;\; \ldots \;\; q_r(x) \right]^T$$

$Z(x)$ is assumed to be a stochastic function with covariance given as

$$\mathrm{COV}[Z(x^{(i)}), Z(x^{(j)})] = \sigma_z^2 R_{ij}$$

where R_{ij} is the correlation function, typically a modified RBF:

$$R(x^{(i)}, x^{(j)}) = R_{ij} = \exp\left\{ -\sum_{k=1}^{n} \theta_k \left(x_k^{(i)} - x_k^{(j)} \right)^2 \right\}$$

Kriging

Model definition

$$s(x) = \hat{\beta}^T q(x) + \overline{r}(x)^T R^{-1}\left(F - Q\hat{\beta}\right)$$

where

$$Q = \left[\, q(x^{(1)}) \;\; q(x^{(2)}) \;\; \ldots \;\; q(x^{(N)}) \,\right]^T_{N\times r}$$

$$\overline{r} = \left[\, R(x^{(1)}, x) \quad R(x^{(2)}, x) \;\; \ldots \;\; R(x^{(N)}, x) \,\right]^T$$

$$\hat{\beta} = \left(Q^T R^{-1} Q\right)^{-1} Q^T R^{-1} F$$

$$F = \left[\, f(x^{(1)}) \;\; f(x^{(2)}) \;\; \ldots \;\; f(x^{(N)}) \,\right]^T$$

Kriging

Correlation parameters θ_k are obtained as

$$\max_{\theta} \left\{ -\frac{1}{2} \left[p \ln(\hat{\sigma}_z^2) + \ln(\det(R)) \right] \right\}$$

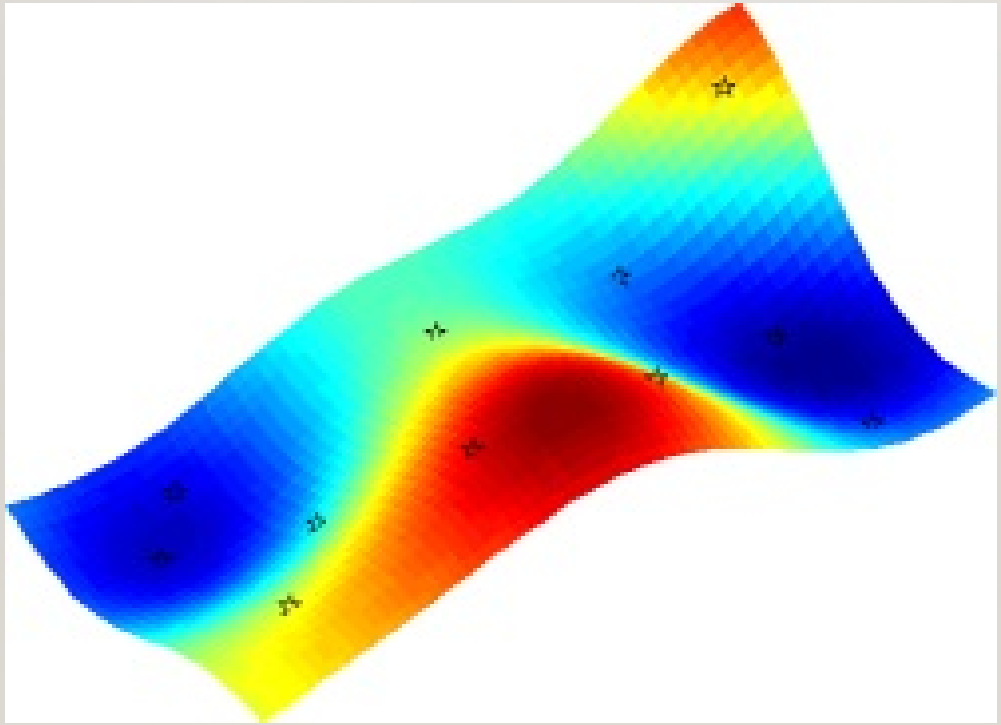
where

$$\hat{\sigma}_z^2 = \frac{1}{p} \left(F - Q\hat{\beta} \right)^{-1} R^{-1} \left(F - Q\hat{\beta} \right)$$

(both $\hat{\sigma}_z^2$ and R depend on θ)

DACE – A Matlab Kriging Toolbox

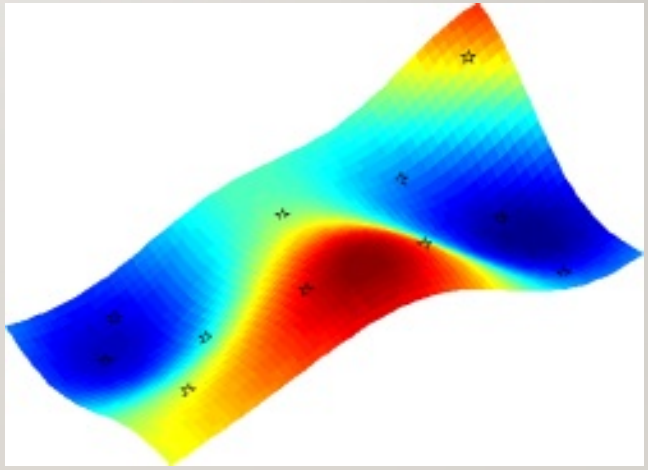
DACE is a freely available Matlab toolbox for kriging approximation by S.N. Lophaven, H.B. Nielsen, and J. Sondergaard from Technical University of Denmark



DACE – A Matlab Kriging Toolbox

DACE is a freely available Matlab toolbox for kriging approximation by S.N. Lophaven, H.B. Nielsen, and J. Sondergaard from Technical University of Denmark

DACE allows identifying and using kriging models based on data pairs provided by the user; it supports polynomial regression models of orders 0, 1, and 2 (as a trend function), as well as a number of correlation models including linear, exponential, and Gaussian ones



DACE also provides some design of experiments utility, including rectangular grids as well as Latin hypercube sampling

DACE – A Matlab Kriging Toolbox

DACE is available for download from the following location:
http://www2.imm.dtu.dk/projects/dace/

Using DACE: Model Construction

The kriging model is found using the function *dacefit*, which has the following syntax:

```
dmodel = dacefit(X,Y,regr,corr,theta0)
```

or

```
dmodel = dacefit(X,Y,regr,corr,theta0,lb,ub)
```

where input arguments are:

X	base points ($N \times n$ array with X(i,:) = $(x^{(i)})^T$)
Y	responses at X ($N \times q$ array with responses at X)
regr	handle to a trend (regression) function
corr	handle to a correlation function
theta0	if lb/ub are not present, theta0 should be the correlation function parameters θ, otherwise theta0 is an initial guess on θ
lb, ub	lower and upper bounds on θ (optional)

Using DACE: Model Construction

Output argument *dmodel* is the structure containing the kriging model, in particular, handles to the regression and correlation functions, correlation parameters, etc.

Supported regression functions

regpoly0	zero order polynomial
regpoly1	first order polynomial
regpoly2	second order polynomial

Selected supported correlation functions $R(\theta, y, x) = \prod_{j=1}^{n} R_j(\theta_j, y_j - x_j)$

correxp	$R_j(\theta_j, y_j, x_j) = \exp\left(-\theta_j \mid y_j - x_j \mid\right)$
corrgauss	$R_j(\theta_j, y_j, x_j) = \exp\left(-\theta_j (y_j - x_j)^2\right)$
corrlin	$R_j(\theta_j, y_j, x_j) = \max\{0, 1 - \theta_j \mid y_j - x_j \mid\}$

Using DACE: Model Evaluation

The DACE kriging model can be evaluated using the function *predictor*, which has the following syntax:

```
y = predictor(x, dmodel)
```

where input arguments

x	trial points ($K \times n$ array with x(i,:) = $(x^{(i)})^T$)
dmodel	structure with the DACE model

and output argument

y	model response ($K \times q$ array with responses at x)

Using DACE: Design of Experiments

DACE provides routines implementing two design of experiments approaches:

1. Rectangular grid: `X = gridsamp(range,q)`

   where *range* is $2 \times n$ array with lower and upper limits, and *q* is a vector holding the number of intervals in subsequent directions

2. Latin hypercube sampling: `X = lhsamp(N,n)`

   where *N* is number of sample points, and *n* is a number of dimensions

In both cases, *X* is $N \times n$ array with the generated sample points (in case of Latin hypercube sampling, the points are normalized to the unit interval

Using DACE: Example

Create a kriging model for the function *f* defined as:

$$f(x) = 3\exp\left(-\frac{\| x - y^{(1)} \|^2}{2}\right) + 4\exp\left(-\frac{\| x - y^{(2)} \|^2}{2}\right) + 2\exp\left(-\frac{\| x - y^{(3)} \|^2}{2}\right)$$

with $y^{(1)} = [1\ 2]^T$, $y^{(2)} = [-2\ 1]^T$, and $y^{(3)} = [0\ -2]^T$. The domain of interest is [-3,3]×[-3,3]. Use Latin hypercube sampling, zero-order polynomial as a trend function and Gaussian correlation function.

Make the surface plot of *f* and the surface plot of the kriging model; indicate base points as black dots.

Repeat the task for N = 10, 20, 50 and 100

Using DACE: Example

Implementation of function *f*:

```
function f=three_peaks(x)
m=2;
f=3*exp(-norm(x-[1 2]')^2/m)+4*exp(-norm(x-[-2 1]')^2/m)+2*exp(-norm(x-[0 -2]')^2/m);
```

Design of experiments and creating the model:

```
n=2;
N=10;                % 20, 50, 100
X=lhsamp(N,n);       % generate samples
lb=[-3 -3];          % lower bound
ub=[3 3];            % upper bound
for j=1:N
   X(j,:)=lb+(ub-lb).*X(j,:); % denormalize sample points
   Y(j)=feval('three_peaks',X(j,:)'); % evaluate fun at sample points
end
dmodel=dacefit(X,Y,@regpoly0,@corrgauss,[1 1],[0.01 0.01],[10 10]); % create model
```

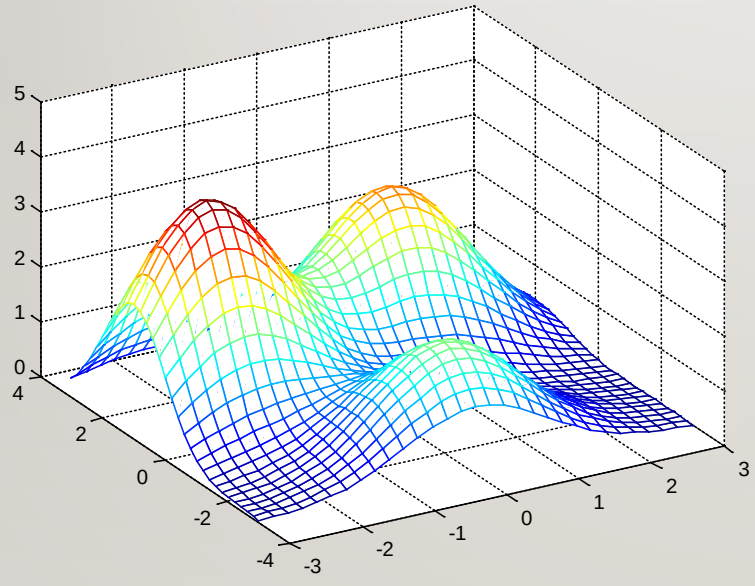
Using DACE: Example

Evaluation of the kriging model and plots:

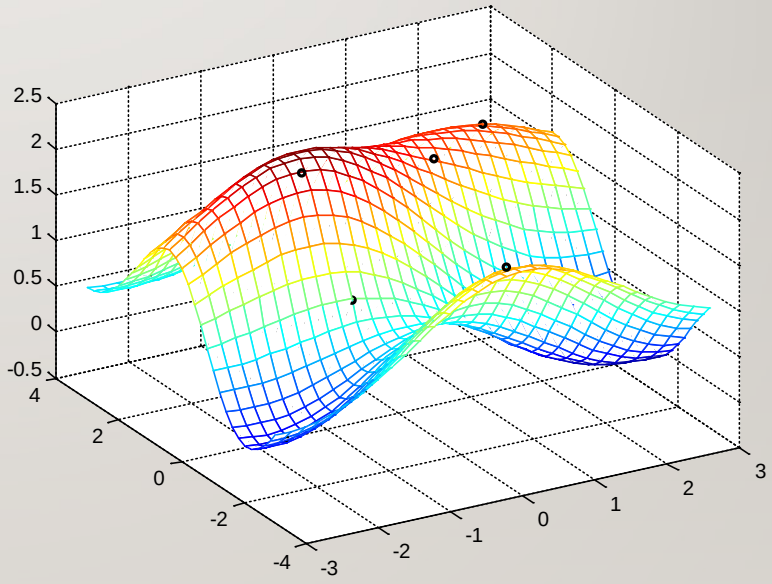
```
xg=-3:0.2:3;
yg=-3:0.2:3;
[xi,yi]=meshgrid(xg,yg);
for j=1:length(xg)
   for k=1:length(yg)
      zi(k,j)=feval('three_peaks',[xg(j) yg(k)]');     % evaluate function f
      vi(k,j)=predictor([xg(j) yg(k)],dmodel);         % evaluate model
   end
end
mesh(xi,yi,zi);                                        % surface plot of the function f
view([-30 35]);
figure;
mesh(xi,yi,vi);                                        % surface plot of the kriging model
hold on
for j=1:N
   plot3(S(j,1),S(j,2),Y(j),'ko','LineWidth',2,'MarkerSize',4); % plot of sample points
end
view([-30 35]);
hold off
```

Using DACE: Example

Results for $N = 10$:



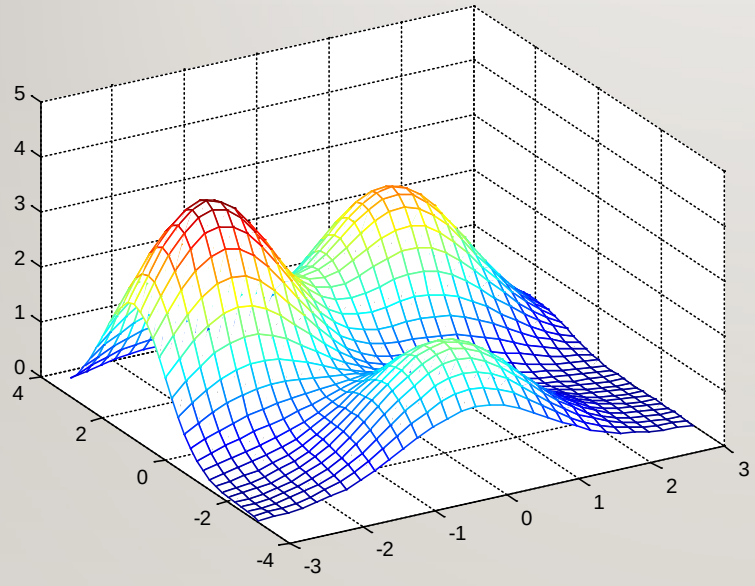
Function f



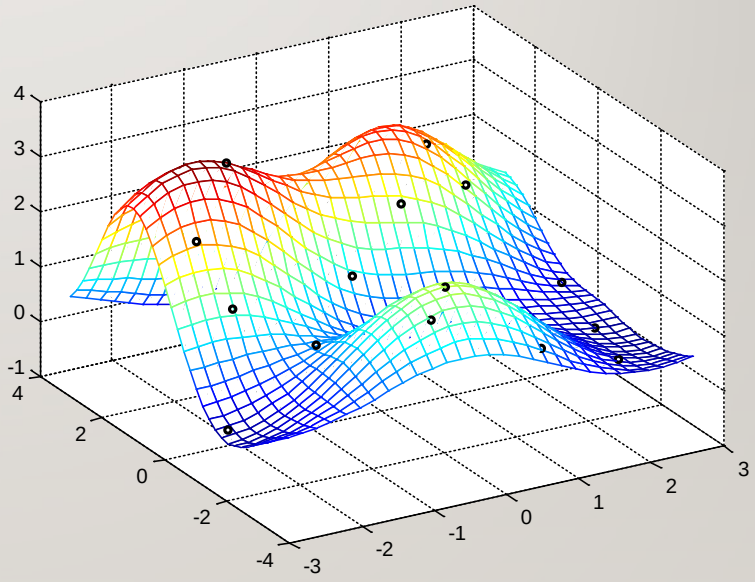
Kriging model

Using DACE: Example

Results for $N = 20$:



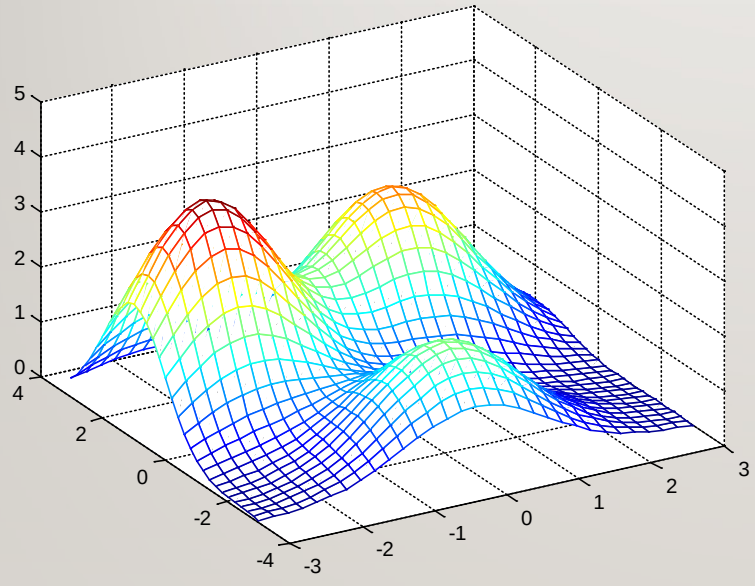
Function f



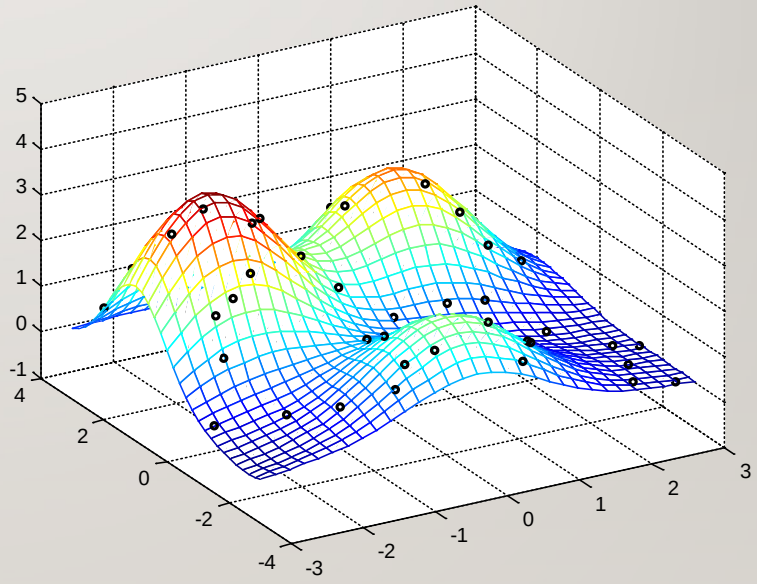
Kriging model

Using DACE: Example

Results for $N = 50$:



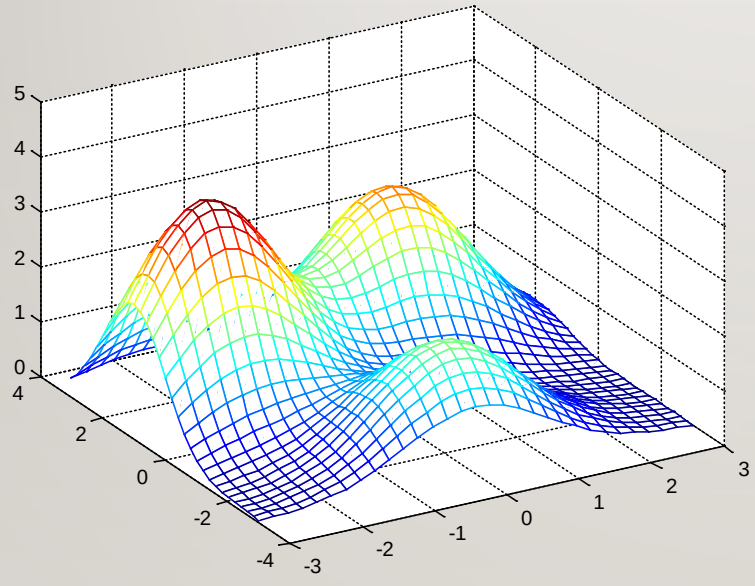
Function f



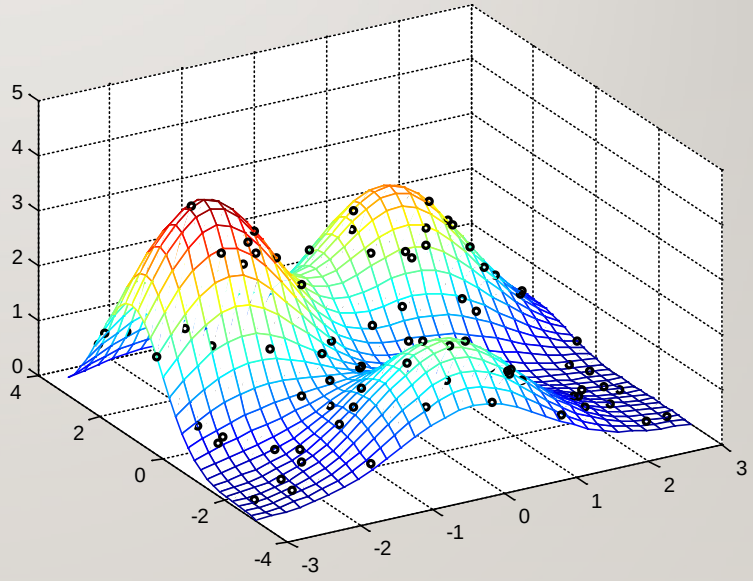
Kriging model

Using DACE: Example

Results for $N = 100$:



Function f



Kriging model

Neural Networks

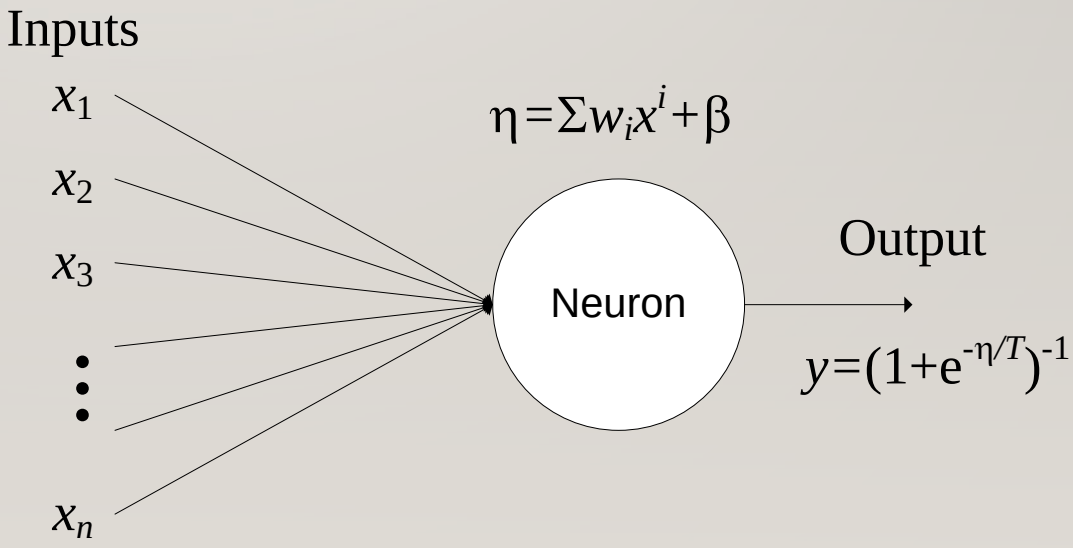
Neural networks are nonlinear regression models

A neural network is composed of neurons (single-unit perceptrons)

If the inputs to each neuron are denoted $\{x^1, x^2, ..., x^n\}$, and the regression coefficients are denoted by the weights w_i, then the output y, might be given by

$$y = \frac{1}{1 + e^{-\eta / T}}$$

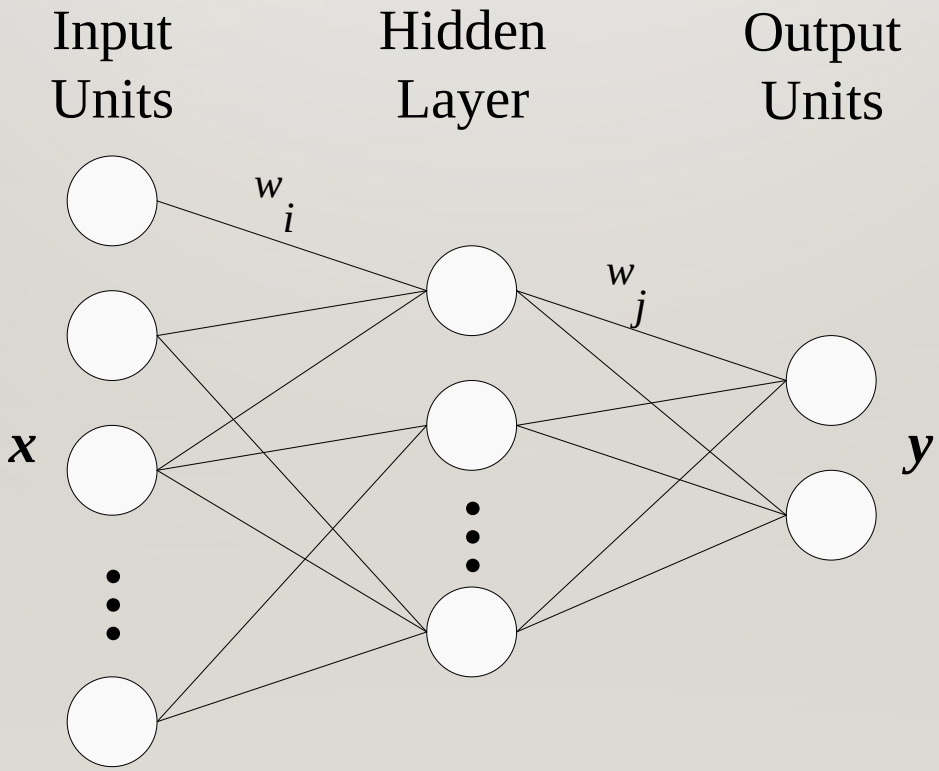
where $\eta = \sum w_i x^i + \beta$ (β is the bias value of a neuron), and T is the slope parameter



Neural Networks

Neural network is created by assembling neurons into an architecture

The most common architecture is multi-layer feedforward network

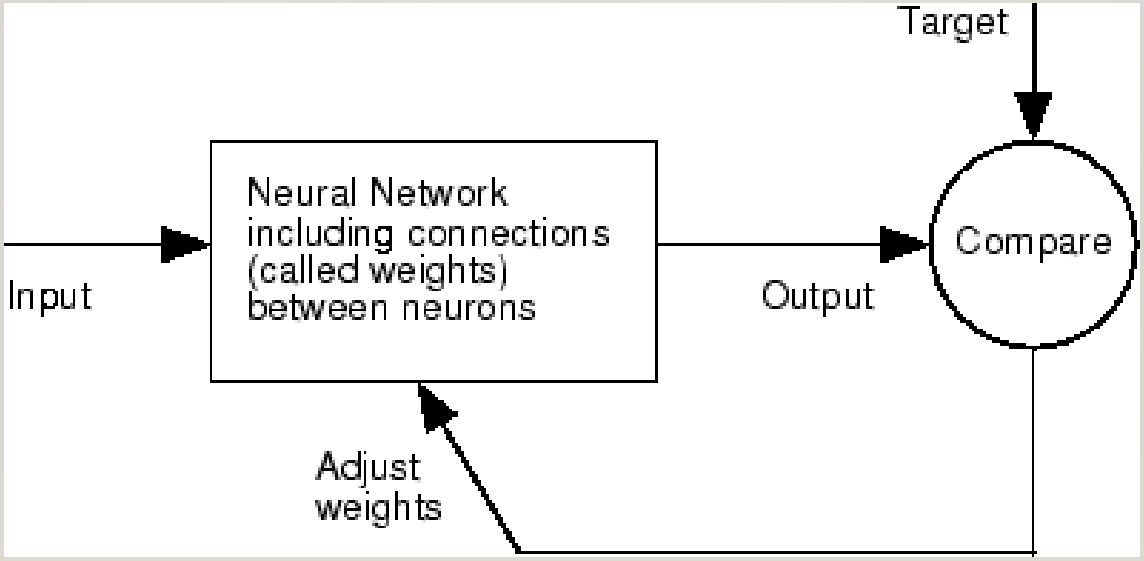


Neural Networks

Building a neural network:

1. Specifying the architecture

2. Training the network to perform well with reference to a training set



Neural Networks

Typical training a neural network: back-propagation
for a set of training data $\{(x^1,y_1), (x^2,y_2), \ldots, (x^p,y_p)\}$ determine the proper values of all weights so that the approximation error

$$E = \sum_j \left(y_j - \hat{y}_j \right)^2$$

is minimized, where $\hat{y}_j$ is the output of the network given input x^j

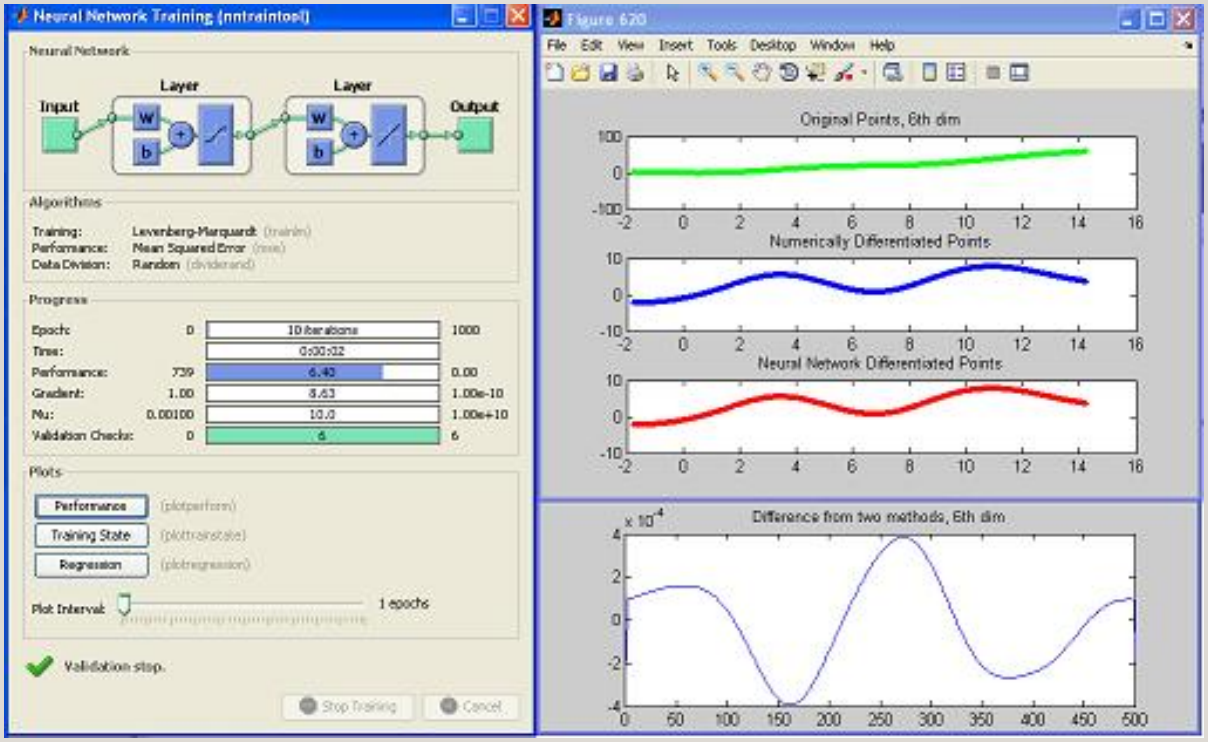
weight are adjusted based on

$$\frac{\partial E}{\partial y} \frac{\partial y}{\partial w_{ij}}$$

Neural Networks Matlab Toolbox

Neural Network Toolbox™ provides algorithms, pretrained models, and apps to create, train, visualize, and simulate both shallow and deep neural networks. You can perform classification, regression, clustering, dimensionality reduction, time-series forecasting, and dynamic system modeling and control.

Videos



Bibliography

T.W. Simpson, J.D. Peplinski, P.N. Koch, and J.K. Allen, “Metamodels for computer-based engineering design: survey and recommendations,” *Engineering with Computers*, vol. 17, no. 2, pp. 129-150, 2001.

N.V. Quiepo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanthan, and P.K. Tucker, “Surrogate-based analysis and optimization,” *Progress in Aerospace Sciences*, vol. 41, pp. 1-28, 2005.

A.A. Mullur, A. Messac, “Metamodeling using extended radial basis functions: a comparative approach,” *Engineering with Computers*, vol. 21, pp. 203-217, 2006.

T.W. Simpson, T.M. Maurey, J.J. Korte, and F. Mistree, “Kriging models for global approximation in simulation-based multidisciplinary design optimization,” *AIAA Journal*, vol. 39, no. 12, pp. 2233-2241, Dec. 2001.

M. Meckesheimer and A.J. Booker, “Computationally inexpensive metamodel assessment strategies,” *AIAA Journal*, vol. 40, no. 10, pp. 2053-2060, Oct. 2002.

W.C.M. van Beers and J.P.C. Kleijnen, “Kriging interpolation in simulation: survey,” *Proc. 2004 Winter Simulation Conf.*, pp. 113-121, Washington, DC, USA, 2004.

R.M. Biernacki, J.W. Bandler, J. Song, and Q.J. Zhang, “Efficient quadratic approximation for statistical design,” *IEEE Trans. Circuits and Systems*, vol.36, pp.1449–1454, 1989.

J. E. Rayas-Sánchez, “EM-based optimization of microwave circuits using artificial neural networks: the state of the art,” *IEEE Trans. Microwave Theory Tech.*, vol. 52, pp. 420-435, Jan. 2004.

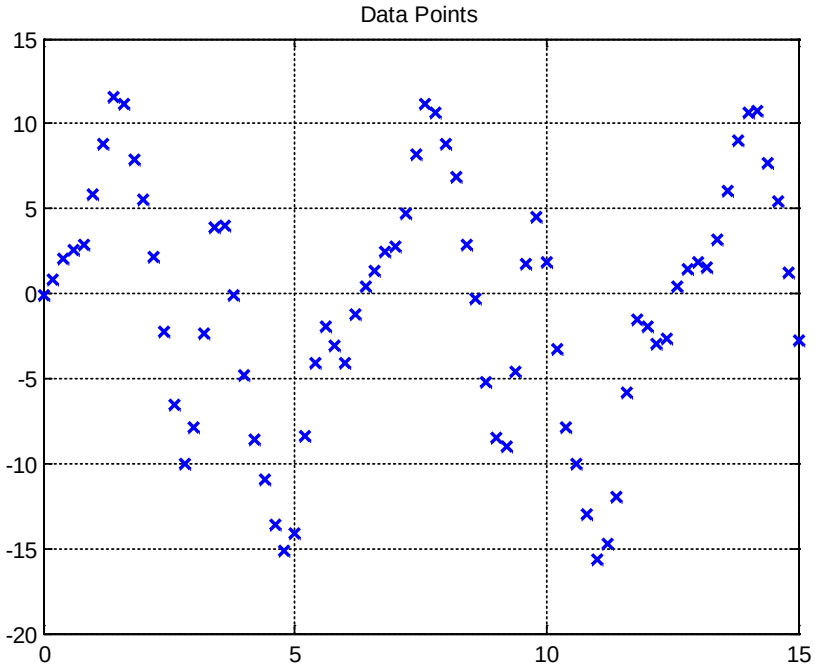
Exercise 1: Linear Regression Models

Consider a function $f(x) = \exp(-x_1/2-x_2/3)$. Perform the following tasks:

1. Evaluate f at the following points: $[0\ 0]^T$, $[0\ 2]^T$, $[0\ 4]^T$, $[2\ 0]^T$, $[2\ 2]^T$, $[2\ 4]^T$, $[4\ 0]^T$, $[4\ 2]^T$, $[4\ 4]^T$.
2. Construct a linear regression model of the form $s_1(x) = \lambda_0 + \lambda_1 x_1 + \lambda_2 x_2$ using the data obtained in point 1.
3. Construct a linear regression model of the form $s_2(x) = \lambda_0 + \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_1 x_2 + \lambda_4 x_1^2 + \lambda_5 x_2^2$ using the data obtained in point 1.
4. Plot function f as well as regression models s_1 and s_2 in the interval $[0\ 4] \times [0\ 4]$ on three separate plots.
5. Add sampled data from point 1 to the surface plots of the regression model.
6. Calculate the least square error of both regression models at base points.

Exercise 2: Linear Regression of Periodic Data

Consider a data pairs provided in a file *exercise_2_data.mat*, and shown in a plot:



Construct a linear regression model of this data using basis functions of the form: 1, sin(x), cos(x), sin(2x), cos(2x), …, sin(nx), cos(nx).

Plot both the input data and the regression function for different values of n. Plot approximation error versus n.

What is the smallest n ensuring sufficient accuracy of the regression model in this case?

Exercise 3: Nonlinear Regression Models

Consider the following data pairs:

0.0000	2.9004	2.0000	1.0805	4.0000	0.7995	6.0000	0.6602	8.0000	0.0669
0.2000	2.8718	2.2000	0.9076	4.2000	0.8182	6.2000	0.6062	8.2000	0.0647
0.4000	2.6121	2.4000	0.7522	4.4000	0.9791	6.4000	0.4935	8.4000	0.0174
0.6000	2.5500	2.6000	0.7779	4.6000	0.9631	6.6000	0.3788	8.6000	0.0864
0.8000	2.3605	2.8000	0.6789	4.8000	1.0600	6.8000	0.2423	8.8000	0.0424
1.0000	2.0048	3.0000	0.6358	5.0000	1.1088	7.0000	0.1860	9.0000	0.0190
1.2000	1.8463	3.2000	0.5275	5.2000	1.1188	7.2000	0.1158	9.2000	0.0805
1.4000	1.5930	3.4000	0.5860	5.4000	1.0386	7.4000	0.1396	9.4000	0.0670
1.6000	1.2475	3.6000	0.6809	5.6000	0.9028	7.6000	0.1260	9.6000	0.0761
1.8000	1.1892	3.8000	0.7591	5.8000	0.8256	7.8000	0.1131	9.8000	0.0242
								10.0000	0.0561

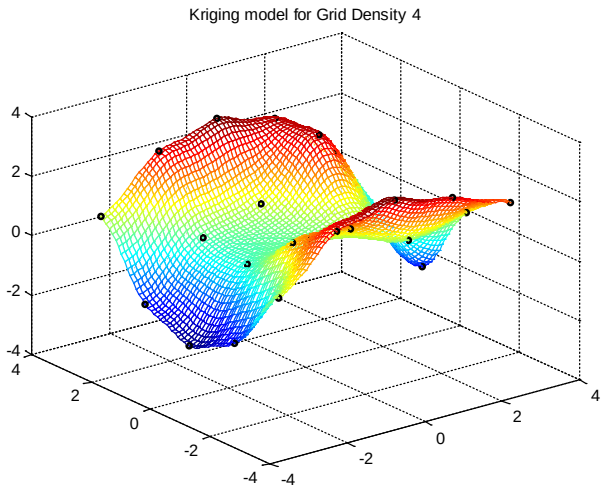
1.Plot the data and think of possible nonlinear regression model that has no more than five parameters.
2.Implement the model and obtain its parameters using *lsqnonlin* routine from Matlab Optimization Toolbox.
3.Plot both the input data and the regression function.

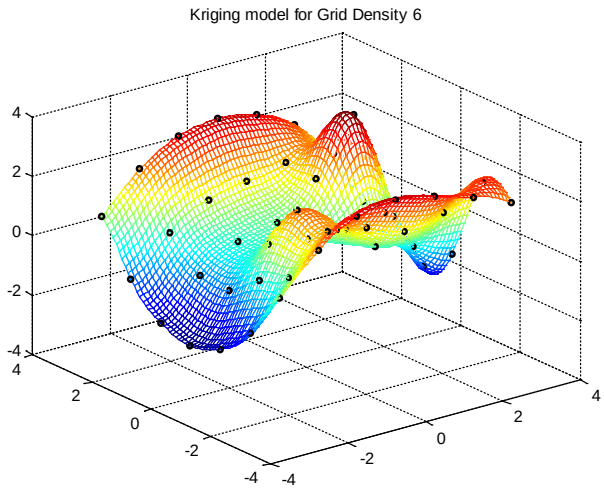
Exercise 4: Radial Basis Functions

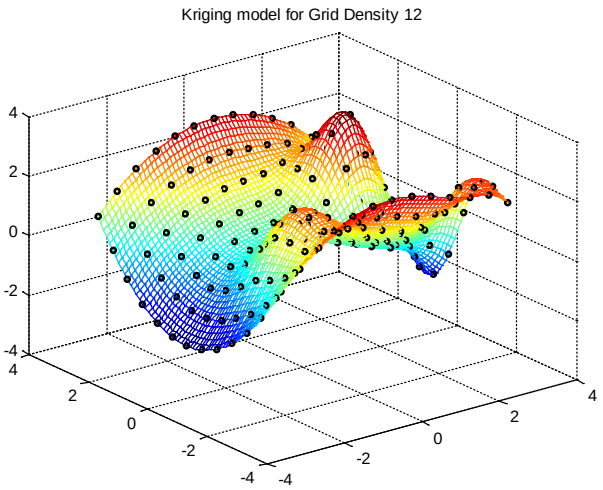
Create a radial basis function surrogate model for function *exercise_4_function*. The domain is [-3,3]×[-3,3]. Use uniform-grid base set with N = 9, 25, 49, 100, and 169 points. Use Gaussian basis functions with c = 1.

Make the surface plot of f and the surface plot of the RBF model; indicate base points as black dots. Calculate the approximation error for each N using a separate set of 100 test points generated using LHS.

Plots of the RBF surrogate for N = 25, 49, and 169:







Exercise 5: Kriging

Use DACE package to create a kriging model for the function f of variables x, y, z, and a free parameter t, $f(x,y,z;t)$, implemented as *exercise_5_function*.The domain of interest is $[0.5\ 1.5] \times [0.5\ 1.5] \times [0.5\ 1.5]$, the values for t are fixed at 0, 0.1, 0.2, …, 1.9, 2.0. The function should be considered as a vector-valued function with components $[f(x,y,z,0.0)\ \ f(x,y,z,0.1)\ \ \ldots\ \ f(x,y,z,2.0)]$.

Use LHS, zero-order polynomial as a trend function and Gaussian correlation function. Select 10 random test points and plot both the function and the kriging model for all points on one plot. Repeat the task for different number of samples N = 20, 50, 100, 200 and 500. Calculate the approximation error for each N.

Plots of the function and kriging model for N = 50, 100 and 500: